



King's Research Portal

DOI:

[10.1093/logcom/ext037](https://doi.org/10.1093/logcom/ext037)

Document Version

Peer reviewed version

[Link to publication record in King's Research Portal](#)

Citation for published version (APA):

Clark, A. (2013). The syntactic concept lattice: Another algebraic theory of the context-free languages? *Journal of Logic and Computation*, N/A(N/A), N/A. [N/A]. <https://doi.org/10.1093/logcom/ext037>

Citing this paper

Please note that where the full-text provided on King's Research Portal is the Author Accepted Manuscript or Post-Print version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version for pagination, volume/issue, and date of publication details. And where the final published version is provided on the Research Portal, if citing you are again advised to check the publisher's website for any subsequent corrections.

General rights

Copyright and moral rights for the publications made accessible in the Research Portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognize and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the Research Portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the Research Portal

Take down policy

If you believe that this document breaches copyright please contact librarypure@kcl.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

The Syntactic Concept Lattice

Another algebraic theory of the context-free languages?

Alexander Clark

Received: date / Accepted: date

Abstract The syntactic concept lattice is a residuated lattice associated with a given formal language; it arises naturally as a generalisation of the syntactic monoid in the analysis of the distributional structure of the language. In this paper we define the syntactic concept lattice and present its basic properties, and its relationship to the universal automaton and the syntactic congruence; we consider several different equivalent definitions, as Galois connections, as maximal factorisations and finally using universal algebra to define it as an object that has a certain universal (terminal) property in the category of complete idempotent semirings that recognize a given language, applying techniques from automata theory to the theory of context-free grammars. We conclude that grammars that are minimal, in a certain weak sense, will always have nonterminals that correspond to elements of this lattice, and claim that the syntactic concept lattice provides a natural system of categories for representing the denotations of context-free grammars.

1 Introduction

The theory of finite automata as representations of regular languages is now very well developed; there are three types of canonical representations. First the canonical minimal deterministic finite automaton DFA, based on the Myhill-Nerode congruence; secondly, the universal automaton (UA), [Con71,LS⁺08] and finally, the canonical residual automaton [DLT04] occupies a position between these two, being nondeterministic but quite close to the DFA. All of these are based in some sense on the relation between prefixes and suffixes: two strings p and s are related if their concatenation is in the language. Formally, we might write $p \sim_L s$ iff $ps \in L$. The Myhill-Nerode congruence says

Department of Philosophy,
King's College London,
The Strand,
London WC2R 2LS E-mail: alexsclark@gmail.com

that two prefixes are identical if they are related to exactly the same set of suffixes; this gives a partition of the set of prefixes. On the other hand, the UA is based on the Galois connection between prefixes and suffixes, though the standard presentations do not present it in this way. This gives, rather than a partition, an overlapping hierarchy, a lattice, whose automata theoretic interpretation is therefore naturally nondeterministic.

When we move to the theory of context free languages and grammars, we need to consider rather the relation between contexts and substrings, where a context is an ordered pair of strings or, equivalently, a string with a gap in it. We might write this as $(l, r) \sim_L u$ iff $lur \in L$. In this case we have two relevant structures: the syntactic congruence, which corresponds exactly to the Myhill-Nerode congruence, and the recently introduced syntactic concept lattice [Cla11, Cla10b, Cla10c, Cla12] that corresponds to the universal automaton. This is very closely related to the syntactic semiring [Pol01, Pol03]; indeed for regular languages they are isomorphic.¹ The syntactic congruence has been classically studied (e.g. [RS59]) and has recently given rise to a number of algorithms for the grammatical inference of context-free grammars [CE07, Cla10a] that are analogous to the results for regular inference using deterministic finite automata (DFAs) obtained by Angluin [Ang82, Ang87]. We note also the result of Bollig et al for learning residual automata [BHKL09]. However, whereas with regular languages, the restriction to deterministic automata does not reduce the class of languages that can be expressed,² with context-free grammars the situation is quite different. The class of languages that can be expressed using nonterminals that correspond to congruence classes under the syntactic congruence is a subclass of context free languages, which while including all the regular languages, does not include such simple languages as the palindrome language over a two letter alphabet.

In this paper we define the syntactic concept lattice, and give its basic properties, focusing on its relationships to the syntactic congruence and the universal automaton. We also demonstrate that it has a desirable property analogous to that of the universal automaton: namely that it contains a morphic image of every context free grammar for the language. Alternatively, we show that it is the unique (up to isomorphism) smallest structure that interprets a CFG for the language; in other words it is the smallest and simplest object that provides a semantics for the grammar. We can view it as the syntactic monoid lifted from monoids (semigroups with unit) to semirings; as lifting from strings to sets of strings. Figure 1 shows the relationship between these structures.

In Section 2 we will define our notation, and define the basic algebraic structures we use: monoids, and complete idempotent semirings, as well as the basic notions of context-free grammars. Then in Section 3 we define the syntactic concept lattice, and explain its basic properties. We then, in Section 4

¹ The syntactic semiring is a quotient of the set of finite sets of strings.

² It is worth recalling here that the size of the minimal DFA for a given language may be exponentially larger than the size of the minimal nondeterministic automaton for the same language, so the two representations are not entirely equivalent.

	$p \sim s$	$(l, r) \sim u$
Equivalence relation	Myhill-Nerode relation	Syntactic congruence
Lattice	Universal automaton	Syntactic concept lattice

Table 1 Relationship between congruences and lattices when considering regular grammars (on the left) and context free grammars on the right.

start to develop an algebraic theory for the semantics of context free grammars, algebraic structures that we use to interpret them. We then define the basic ideas of syntactic morphisms – i.e. morphisms of CFGs in Section 5, which leads us onto the universal morphism using the Syntactic Concept Lattice (Section 6).

We then look briefly at the theory of regular languages in Section 7, discussing the Universal Automaton and its relationship(s) to the Syntactic Concept Lattice. We then conclude with some discussion of the relevance of this work for representational assumptions in linguistics.

2 Preliminaries

We use standard set theoretic notation; the empty set is \emptyset , and $\subseteq, \supseteq, \cup$ and \cap have their normal meanings. We write $\mathcal{P}(X)$ for the power set of X , and $\mathcal{F}(X)$ for the collection of only finite subsets of X .

We have a finite nonempty set Σ which we call an alphabet. Let Σ^* be the set of all finite length strings over Σ . We write λ for the empty string. We will use letters at the beginning of the alphabet, a, b, c to refer to elements of Σ and abusing notation slightly also to strings of length 1. We write $|u|$ for the length of a string $u \in \Sigma^*$; so $|\lambda| = 0$. We will write $|u|_a$ for the number of times the element $a \in \Sigma$ occurs in the string u . We will write concatenation of strings u, v as uv and sometimes $u \cdot v$ when we want to be explicit.

A language is any subset of Σ^* . For two such sets A, B we define the product $AB = \{uv \mid u \in A, v \in B\}$. A context is an ordered pair of strings $(l, r) \in \Sigma^* \times \Sigma^*$. We can combine a context (l, r) and a string u using the wrapping operation which we write $(l, r) \odot u = lur$. (λ, λ) is the empty context; clearly $(\lambda, \lambda) \odot u = u$ for any u . We will lift these operations to sets in the standard way. If C is a set of contexts, and S is a set of strings, then $C \odot S = \{lur \mid (l, r) \in C, u \in S\}$. If X, Y are sets of strings then $X \times Y$ is the set of contexts $\{(x, y) \mid x \in X, y \in Y\}$. We also extend the wrap operation \odot to contexts as $(x, y) \odot (p, q) = (xp, qy)$, so that $(x, y) \odot (p, q) \odot u$ is associative, and to sets of contexts in the natural way.

2.1 Monoids

We use a few standard algebraic structures; in order to make this self contained we define all the basic properties here; we will sometimes be rather casual about the difference between an algebraic structure, and its reducts and expansions.

A monoid is a set M together with a unit, 1_M and an associative operation \circ_M . A monoid morphism from $\langle M, 1_M, \circ_M \rangle$ to $\langle N, 1_N, \circ_N \rangle$ is a function $h : M \rightarrow N$ such that $h(a \circ_M b) = h(a) \circ_N h(b)$ and $h(1_M) = 1_N$. Note then that $\langle \Sigma^*, \lambda, \cdot \rangle$ is a monoid, the free monoid over Σ ; the operation is simply concatenation.

We write M_1 for the trivial one element monoid.

2.2 Semirings

Rather than just a monoid (a semigroup with unit) we will consider a slightly more complicated structure which in addition to the concatenation operation \circ has another operation which we will write as \vee .

Definition 1 An idempotent semiring is a structure which we write as $\langle M, \vee, \circ, \perp, 1 \rangle$ where

- $\langle M, \vee, \perp \rangle$ is a commutative idempotent monoid with \perp as unit; i.e. $x \vee \perp = x = \perp \vee x$. Idempotent means that $x \vee x = x$.
- $\langle M, \circ, 1 \rangle$ is a monoid
- $\perp \circ x = x \circ \perp = \perp$.
- \circ distributes over \vee , so $a \circ (b \vee c) = (a \circ b) \vee (a \circ c)$ and $(b \vee c) \circ a = (b \circ a) \vee (c \circ a)$.

The free idempotent semiring over Σ is the collection of finite sets of strings of Σ , $\mathcal{F}(\Sigma^*)$, where $1 = \{\lambda\}$, $\perp = \emptyset$, $\vee = \cup$ and $X \circ Y = XY$ is concatenation extended to sets of strings.

We need to add some infinitary operations here – in particular we want to be able to take the \vee (the union, intuitively) over infinite sets. This structure is then a complete idempotent semiring [Eil74, Gol99, ÉL05].

Definition 2 A complete idempotent semiring (CIS) is an idempotent semiring $\langle M, \vee, \circ, \perp, 1 \rangle$ where additionally

- \vee is defined over arbitrary sets. Given a family of sets $\{x_i \mid i \in I\}$, there is an element $\bigvee_{i \in I} x_i$.
- \circ distributes over \vee : if X is a collection of elements then $a \circ (\bigvee_{i \in I} x_i) = \bigvee_{i \in I} a \circ x_i$ and $(\bigvee_{i \in I} x_i) \circ a = \bigvee_{i \in I} (x_i \circ a)$

The free structure generated by this over Σ is $\mathcal{P}(\Sigma^*)$.

The elements of this are finite or infinite sets of finite length strings over Σ . If we had just an idempotent semiring, the free structure would be the collection of finite sets of strings – since the languages we use might be infinite, we need to allow infinite sets and therefore a complete semiring.

These CIS have a natural partial order.

Proposition 1 Given a CIS, we can define a relation \leq by $X \leq Y$ iff $Y = X \vee Y$. This relation is a partial order.

Complete idempotent semirings are the appropriate structures for interpreting context-free grammars as systems of equations [GR62, Lei12].³

Note that a CIS is an ω -complete partial order. Every infinite ascending chain a_1, a_2, \dots , where $a_i \leq a_{i+1}$, has a least upper bound, which is equal to $\bigvee_i a_i$.

2.3 Residuated lattices

Definition 3 A complete lattice is a structure $\langle S, \leq \rangle$ where every subset A of S has a greatest lower bound and least upper bound denoted by $\bigwedge A$ and $\bigvee A$. In such a lattice we can define binary operations \vee, \wedge on S by $x \vee y = \bigvee \{x, y\}$ and $x \wedge y = \bigwedge \{x, y\}$, and top element $\top = \bigvee S$, and bottom element $\perp = \bigwedge S$.

Definition 4 [GJKO07] A complete residuated lattice is a complete lattice $\langle S, \leq \rangle$ with three additionally binary operations $\circ, /, \backslash$ and a unit 1 where $\langle S, 1, \circ \rangle$ is a monoid and the three binary operations satisfy the identities, for all $X, Y, Z \in S$

$$Y \leq X \backslash Z \text{ iff } X \circ Y \leq Z \text{ iff } X \leq Z / Y$$

Proposition 2 [Jip04] Any complete idempotent semiring can be expanded to a complete residuated lattice by defining:

$$\begin{aligned} X \wedge Y &= \bigvee \{Z \mid Z < X, Z < Y\} \\ X / Y &= \bigvee \{Z \mid Z \circ Y \leq X\} \\ X \backslash Y &= \bigvee \{Z \mid X \circ Z \leq Y\} \end{aligned}$$

2.4 Congruences

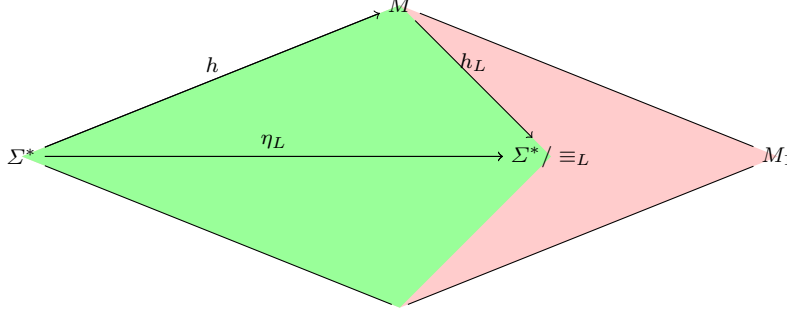
A congruence on an algebra is an equivalence relation on the elements that respects the operations of the algebra. A congruence of a monoid then is an equivalence relation \cong such that if $u \cong u'$ and $v \cong v'$ then $u \circ v \cong u' \circ v'$. We write $[u]_{\cong}$ for the equivalence class of u .

The equivalence classes under a congruence form an algebra where $[1]_{\cong}$ is a unit, and the concatenation operation is $[u]_{\cong} \circ [v]_{\cong} = [uv]_{\cong}$. This is well defined since it is a congruence.

A homomorphism is analogously a function between algebras which respects the operations. The congruences and the surjective homomorphisms are closely related. Given a surjective homomorphism h from $A \rightarrow B$, the equivalence relation on A given by $u \cong v$ iff $h(u) = h(v)$ is a congruence. Conversely, given a congruence \cong , the natural map $h : u \rightarrow [u]_{\cong}$ is a surjective homomorphism.

³ I am grateful to Hans Leiss for making this point.

Fig. 1 The syntactic congruence: the coarsest congruence that recognises L . Fine congruences are on the left and on the far right is the coarsest congruence of all with only one congruence class; the quotient monoid is the trivial one element monoid M_1 . Congruences that recognise L are shaded green. Congruences that do not are shaded red. If M is a monoid that recognises L through the morphism h then there is a homomorphism from M to Σ^* / \equiv_L , h_L such that $\eta_L = h \cdot h_L$.



Given a free algebra, like the free monoid Σ^* , the congruences form a lattice; and therefore also do the corresponding quotient monoids. The bottom element is Σ^* , the finest congruence, and the top is the coarsest congruence where everything is equivalent to everything else and there is only one element in the quotient. For any surjective homomorphism from Σ^* onto a monoid A , we can identify a congruence which is an element of this lattice. Similarly if we have two congruences, one of which is finer than the other, then there is a surjective homomorphism from the finer to the coarser.

The same applies, as we shall see, for congruences and homomorphisms of complete idempotent semirings.

2.5 The syntactic congruence

The syntactic concept lattice, which is the main topic of this paper, is best understood in relation to the syntactic congruence [Eil74]; therefore, we recall the elementary properties of the syntactic congruence in a notation consistent with the rest of this paper.

Definition 5 For a language L and $w \in \Sigma^*$; define the distribution of w to be:

$$C_L(w) = \{(l, r) \in \Sigma^* \times \Sigma^* \mid lwr \in L\}$$

Definition 6 We say that two strings $u, v \in \Sigma^*$ are congruent w.r.t. a language L , written $u \equiv_L v$ iff $C_L(u) = C_L(v)$.

Proposition 3 \equiv_L is an equivalence relation and a congruence of the monoid Σ^* ; i.e. $u \equiv_L v$ implies that for any $z \in \Sigma^*$, $uz \equiv_L vz$ and $zu \equiv_L zv$.

Definition 7 We write $[u]_L$ for the equivalence class of u under \equiv_L .

$$[u]_L = \{v \in \Sigma^* \mid u \equiv_L v\}$$

Proposition 4 For any language L and any strings, u, v

$$[uv]_L \supseteq [u]_L[v]_L$$

The following proposition is classical, attributed to Myhill [RS59].

Proposition 5 The number of congruence classes of L is finite iff L is regular.

Example 1 Suppose $L = \Sigma^*$. Then there is one congruence class which consists of all strings.

Example 2 Consider the regular language $L = (ab)^*$. This has the following 6 congruence classes which are all regular sets. $(ab)^+$, $\{\lambda\}$, $a(ba)^*$, $b(ab)^*$, $(ba)^+$, and finally all other strings (i.e. not in the preceding 5 classes) which we can express using the regular expression: $\Sigma^*bb\Sigma^* \cup \Sigma^*aa\Sigma^*$.

Example 3 Consider the language $O_1 = \{w \in \{a, b\}^* \mid |w|_a = |w|_b\}$. Define $c(w) = |w|_a - |w|_b$. Clearly $[u]_L = \{v \mid c(u) = c(v)\}$, and so this has an infinite number of congruence classes indexed by the integers where for $n \in \mathbb{Z}$, $C_n = \{w \mid c(w) = n\}$.

Definition 8 A congruence \sim saturates a language L if $x \sim y$ implies that $x \in L$ iff $y \in L$.

Informally, this just means that all of the classes are either inside or outside the language.

Proposition 6 The syntactic congruence saturates L , and is the coarsest congruence that saturates L .

Proof Suppose \sim is a congruence that saturates L , and suppose $x \sim y$. Take some l, r . Then $lxr \sim lyr$ as it is a congruence. Since \sim saturates L then $lxr \in L$ iff $lyr \in L$. Therefore $x \equiv_L y$ and so \sim is finer than \equiv_L .

Definition 9 Given a monoid M and a morphism h from Σ^* to M , we say that the monoid *recognizes* the language L through h if there is a set $X \subseteq M$ such that $h^{-1}(X) = L$, i.e. such that

$$\{w \in \Sigma^* \mid h(w) \in X\} = L$$

.

Note that if M recognises L through the morphism h , then $h(M)$ also recognizes L through the surjective morphism h . $h(M)$ is a submonoid of M if h is a homomorphism. In this case we are not interested in all of M but only in the submonoid $h(M)$. The rest of M consists of elements that are not related to the language at all. We will therefore restrict our consideration to the case where h is surjective.

Definition 10 The syntactic monoid, Σ^* / \equiv_L is a monoid whose elements are the congruence classes of Σ^* under \equiv_L , concatenation being defined as $[uv] = [u] \circ [v]$, with left and right identity $[\lambda]$.

Note that the operation in the quotient is well-defined, as usual, by the fact that it is a congruence.

Example 4 Consider again the language O_1 . This has a syntactic monoid which is isomorphic to the integers under addition.

The syntactic monoid has a certain universal property [Sak09].

Definition 11 The natural map from $\Sigma^* \rightarrow \Sigma^* / \equiv_L$, written η_L , is $u \rightarrow [u]_L$. This is a surjective monoid homomorphism.

Proposition 7 *The syntactic monoid of L recognises L through the morphism η_L .*

Proof This is perhaps too obvious to merit a detailed proof; but if $u \in L$ and $u \equiv_L v$ then v is in L ; so clearly $L = \bigcup \{[u]_L \mid u \in L\}$.

The syntactic monoid of L is the smallest (coarsest) monoid that recognizes L , and it is unique up to isomorphism.

We use the \cdot symbol to represent function composition here.⁴ If $f : A \rightarrow B$ and $g : B \rightarrow C$ then $f \cdot g : A \rightarrow C$.

Proposition 8 *Suppose L is recognized by the monoid M via the surjective morphism h then there is a morphism h_L from M to the syntactic monoid such that $\eta_L = h \cdot h_L$.*

See the diagram in Figure 1 for an illustration of this.

Proof Define the congruence \equiv_h to be $u \equiv_h v$ iff $h(u) = h(v)$. This is a congruence that saturates L . Define h_L to be $h_L(h(x)) = \eta_L(x)$. Since \equiv_h saturates L , this is well-defined. Suppose x and y are such that $h(x) = h(y)$; then $x \equiv_h y$ which implies $x \equiv_L y$ which implies $\eta_L(x) = \eta_L(y)$.

Note that all of this follows quite directly from basic principles of universal algebra: this is an immediate consequence of Noether's third isomorphism theorem, namely that if we have two congruences of an algebra such that one is finer than the other then there is a homomorphism from the quotient of the finer to the quotient of the coarser. We do not rely on this body of mathematics in this paper but prove things directly each time. This leads to a little repetition and a lack of maximal generality, but makes the presentation more self-contained, concrete and comprehensible.

⁴ We do this to avoid confusion with the more standard symbol \circ which here we use merely as concatenation in monoids.

2.6 Context free grammars

A context free grammar (CFG) over an alphabet Σ is a tuple $\langle \Sigma, V, S, P \rangle$, where V is a finite set of nonterminals, $S \in V$ is the unique start symbol, P is a finite set of productions written $N \rightarrow \alpha$ where $N \in V$ and $\alpha \in (V \cup \Sigma)^*$. We define the derivations in the standard way: We define the one step derivation relation by $\beta N \gamma \Rightarrow \beta \alpha \gamma$, when $N \rightarrow \alpha \in P$ and $\beta, \gamma \in (V \cup \Sigma)^*$. We use the notation $\alpha \xRightarrow{*} \beta$ for a derivation using zero or more steps : the reflexive transitive closure of \Rightarrow .

Definition 12 For a CFG, and a nonterminal N we define $\mathcal{L}(G, N) = \{w \in \Sigma^* \mid S \xRightarrow{*} w\}$ and $\mathcal{C}(G, N) = \{(l, r) \in \Sigma^* \times \Sigma^* \mid S \xRightarrow{*} lNr\}$. We also extend $\mathcal{L}(G, \alpha)$ for $\alpha \in (V \cup \Sigma)^*$ in the usual way, with $\mathcal{L}(G, \lambda) = \{\lambda\}$, $\mathcal{L}(G, a) = \{a\}$ for $a \in \Sigma$ and $\mathcal{L}(G, \alpha\beta) = \mathcal{L}(G, \alpha)\mathcal{L}(G, \beta)$.

Note that by this definition $(\lambda, \lambda) \in \mathcal{C}(G, S)$ since $S \xRightarrow{*} S = \lambda S \lambda$, even if S does not occur on the right hand side of any production in G .

$L(G)$ is defined to be $\mathcal{L}(G, S)$, for the unique start symbol S . A context-free language is a language L_* such that there is a CFG G such that $L(G) = L_*$.

Proposition 9 *If we have a grammar G that defines a language L then for any nonterminal N in G ,*

$$\mathcal{C}(G, N) \odot \mathcal{L}(G, N) \subseteq L$$

We will give the trivial proof now.

Proof If $(l, r) \in \mathcal{C}(G, N)$, then there is a derivation $S \xRightarrow{*} lNr$. If $u \in \mathcal{L}(G, N)$, then there is a derivation $N \xRightarrow{*} u$. There is therefore a derivation $S \xRightarrow{*} lNr \xRightarrow{*} lur$, and so $(l, r) \odot u = lur \in L$.

This proposition implies that a nonterminal in a CFG defines a decomposition or factorisation of L into a set of contexts and a set of substrings.

Proposition 10 *If there is a production $N \rightarrow \alpha$, then $\mathcal{L}(G, N) \supseteq \mathcal{L}(G, \alpha)$.*

Proposition 11 *More specifically, if the rule is of the form $N \rightarrow PQ$ then $\mathcal{L}(G, N) \supseteq \mathcal{L}(G, P)\mathcal{L}(G, Q)$. If the rule is of the form $N \rightarrow w$ then $\mathcal{L}(G, N) \supseteq \{w\}$.*

Therefore we have two relations between the grammar and the language it describes. One is that the nonterminal determines a factorisation into a set of contexts and a set of strings. The other is that a production determines an inclusion relation between the sets of strings that correspond to nonterminals. As we shall see later – and in exact analogy to the UA (see e.g. [LS⁺08]) – we can reverse this process and define a grammar where the nonterminals are based on these decompositions – more precisely on the maximal decompositions, and the productions are the ones that satisfy the inclusion relations.

In what follows we will occasionally restrict ourselves to grammars where the productions are either of the form $N \rightarrow PQ$ or $N \rightarrow a$ for $a \in \Sigma$, or $N \rightarrow \lambda$; a restriction along the lines of Chomsky normal form.

3 The Syntactic Concept Lattice

Given any fixed language L , we can define an object called the syntactic concept lattice [Cla11, Cla10b]. This is a canonical object in the sense that it is uniquely defined for each language.

There are several equivalent ways of defining this lattice; we will describe two. The first is based on the Galois connection between sets of contexts and sets of strings [DP02]; this is exactly the same as Formal Concept Analysis [GW97]. We consider some fixed language L in what follows. We define a *polar* map from sets of strings to sets of contexts. For a set of strings S , we define a set of contexts S^\triangleright as follows.

$$S^\triangleright = \{(l, r) \in \Sigma^* \times \Sigma^* \mid \forall u \in S, lur \in L\} \quad (1)$$

This is the shared distribution of the set of strings S . Dually, we define a polar map from a set of contexts C to the set of all strings that can occur in all of these contexts.

$$C^\triangleleft = \{u \in \Sigma^* \mid \forall (l, r) \in C, lur \in L\} \quad (2)$$

We now give some basic properties of these maps.

Proposition 12 *If $S \subseteq T$ are two sets of strings, then $S^\triangleright \supseteq T^\triangleright$. If S is empty, then $S^\triangleright = \Sigma^* \times \Sigma^*$.*

Proposition 13 *If X, Y are sets of strings then $(X \cup Y)^\triangleright = X^\triangleright \cap Y^\triangleright$. Similarly if X_i , for $i \in I$ is a family of sets, finite or infinite, then*

$$\left(\bigcup_{i \in I} X_i \right)^\triangleright = \bigcap_{i \in I} X_i^\triangleright.$$

Dually we have the following two propositions.

Proposition 14 *If $C \subseteq D$ are two sets of contexts, then $C^\triangleleft \supseteq D^\triangleleft$.*

Proposition 15 *If C, D are sets of contexts then $(C \cup D)^\triangleleft = C^\triangleleft \cap D^\triangleleft$. Similarly if C_i , for $i \in I$ is a family of contexts, finite or infinite, then*

$$\left(\bigcup_{i \in I} C_i \right)^\triangleleft = \bigcap_{i \in I} C_i^\triangleleft.$$

Proposition 16 *For any set of strings S , $S \subseteq S^{\triangleright\triangleleft}$. For any set of contexts C , $C \subseteq C^{\triangleleft\triangleright}$.*

Proof Suppose $w \in S$. Then the set of contexts that w can occur in contains S^\triangleright . Therefore $w \in S^{\triangleright\triangleleft}$. Similarly if $(l, r) \in C$, the set of strings that can occur in (l, r) is a superset of C^\triangleleft and so $(l, r) \in C^{\triangleleft\triangleright}$.

Proposition 17 *For any set of strings S , $S^\triangleright = S^{\triangleright\Diamond}$.
For any set of contexts C , $C^\triangleleft = C^{\Diamond\triangleleft}$.*

Proof $(S^\triangleright)^{\triangleright\triangleleft} \supseteq S^\triangleright$ by the previous lemma (putting $C = S^\triangleright$). But $S^{\triangleright\triangleleft} \supseteq S$, so $(S^{\triangleright\triangleleft})^\triangleright \subseteq S^\triangleright$ by Proposition 12. This completes the first proof; the second is exactly parallel.

Definition 13 We say that a set of strings S is *closed* iff $S = S^{\triangleright\triangleleft}$. Similarly, we say that a set of contexts C is *closed* iff $C = C^{\Diamond\triangleleft}$.

Proposition 18 *For any set of strings S , S^\triangleright is closed, and so is $S^{\triangleright\triangleleft}$. For any set of contexts C , C^\triangleleft is closed, and so is $C^{\Diamond\triangleleft}$.*

Proposition 19 *L is always a closed set of strings.*

$(\lambda, \lambda) \in L^\triangleright$. Therefore if $w \in L^{\triangleright\triangleleft}$, w must occur in the context (λ, λ) which implies that $w \in L$.

Proposition 20 *Suppose X and Y are sets of strings; then $(X \cup Y)^{\triangleright\triangleleft} = (X^{\triangleright\triangleleft} \cup Y^{\triangleright\triangleleft})^{\triangleright\triangleleft}$. More generally, if X_i is a family of sets of strings, then $\bigcup_i X_i^{\triangleright\triangleleft} = (\bigcup_i X_i)^{\triangleright\triangleleft}$.*

Proof $(X^{\triangleright\triangleleft} \cup Y^{\triangleright\triangleleft})^\triangleright = X^{\triangleright\Diamond} \cap Y^{\triangleright\Diamond} = X^\triangleright \cap Y^\triangleright = (X \cup Y)^\triangleright$. Similarly $(\bigcup_i X_i^{\triangleright\triangleleft})^\triangleright = \bigcap_i X_i^{\triangleright\Diamond} = \bigcap_i X_i^\triangleright = (\bigcup_i X_i)^\triangleright$.

Definition 14 A *concept* is an ordered pair $\langle S, C \rangle$ of a set of strings S and a set of contexts C such that $S^\triangleright = C$ and $C^\triangleleft = S$.

Though it seems like these concepts might be hard to find, it is easy to prove the following proposition.

Proposition 21 – *For any set of strings S , $\langle S^{\triangleright\triangleleft}, S^\triangleright \rangle$ is a concept.*

– *For any set of contexts C , $\langle C^\triangleleft, C^{\Diamond\triangleleft} \rangle$ is a concept.*

For any set of strings S we define $\mathcal{C}(S)$ to be the concept $\langle S^{\triangleright\triangleleft}, S^\triangleright \rangle$.

Indeed it is easy to see that if $\langle S, C \rangle$ is a concept, then S is a closed set of strings and C is a closed set of contexts. There is a bijection between the set of closed sets of strings and the set of closed sets of contexts; we can either consider the concepts as sets of strings, or sets of contexts, or both.

An alternative way of defining the concepts is as maximal decompositions of a language into sets of strings and contexts..

Definition 15 A concept is an ordered pair $\langle S, C \rangle$ of a set of strings S and a set of contexts C such that $C \odot S \subseteq L$ and S and C are both maximal. That is to say, if $\langle T, D \rangle$ is such that $D \odot T \subseteq L$ and $T \supseteq S, D \supseteq C$ then $T = S$ and $C = D$.

It is easy to verify that these two definitions are equivalent; this second definition makes the links with the universal automaton more explicit.

Example 5 Suppose $L = \Sigma^*$. Then there is exactly one concept: $\langle \Sigma^*, \Sigma^* \times \Sigma^* \rangle$.

Example 6 Suppose $L = \emptyset$. Then there are exactly two concepts: $\langle \Sigma^*, \emptyset \rangle$ and $\langle \emptyset, \Sigma^* \times \Sigma^* \rangle$.

Definition 16 Given a language L we denote the set of concepts of L , by $\mathfrak{B}(L)$. We call this the *syntactic concept lattice* of L .

As we shall see this forms a very rich algebraic structure.

Definition 17 We define a partial order by $\langle S, C \rangle \leq \langle T, D \rangle$ iff $S \subseteq T$.

Definition 18 Suppose $\{\langle S_i, C_i \rangle \mid i \in I\}$ is a collection of concepts. We define $\bigvee \{\langle S_i, C_i \rangle \mid i \in I\} = \langle \bigcup_i S_i, \bigcap_i C_i \rangle$. For two concepts, we define $\langle S_x, C_x \rangle \vee \langle S_y, C_y \rangle = \langle S_x \cup S_y, C_x \cap C_y \rangle$.

It is clear that this is a least upper bound. We use the notation $\mathfrak{B}(L)$ to emphasize the link to formal concept analysis where the fraktur B stands for *Begriff*. We now examine the relationship between the syntactic congruence and the syntactic concept lattice.

Proposition 22 $\{u\}^{\triangleright\triangleleft} = \{v\}^{\triangleright\triangleleft}$ iff $u \equiv_L v$.

Proposition 23 $[u] \subseteq \{u\}^{\triangleright\triangleleft}$.

Proposition 24 If $w \in S$ for some closed set of words S , then $[w] \subseteq S$. In other words, closed sets of strings are unions of congruence classes: $S = \bigcup_{w \in S} [w]$.

Proposition 25 $\mathfrak{B}(L)$ has finitely many elements iff L is regular.

Proof Note that a language L has finitely many congruence classes iff it is regular. By Proposition 22 this means that if L is not regular then $\mathfrak{B}(L)$ is infinite. By Proposition 24, if L is regular, then $\mathfrak{B}(L)$ is finite.

Figure 2 contains a diagram which shows the syntactic concept lattice of the regular language $(ab)^*$. This consists of 7 concepts, which we arrange in a Hasse diagram according to the natural partial order.

Figure 3 contains a similar diagram for a non-regular language, which of course has an infinite number of concepts.

Whereas the syntactic monoid is always countable; and infinite iff the language is regular, the lattice may have uncountably many elements, even for a context-free language; of course the cardinality of the sets can be at most 2^{\aleph_0} , if Σ is finite.

Proposition 26 There is a context-free language L such that $\mathfrak{B}(L)$ has uncountably many elements. (Ryo Yoshinaka, p.c.)

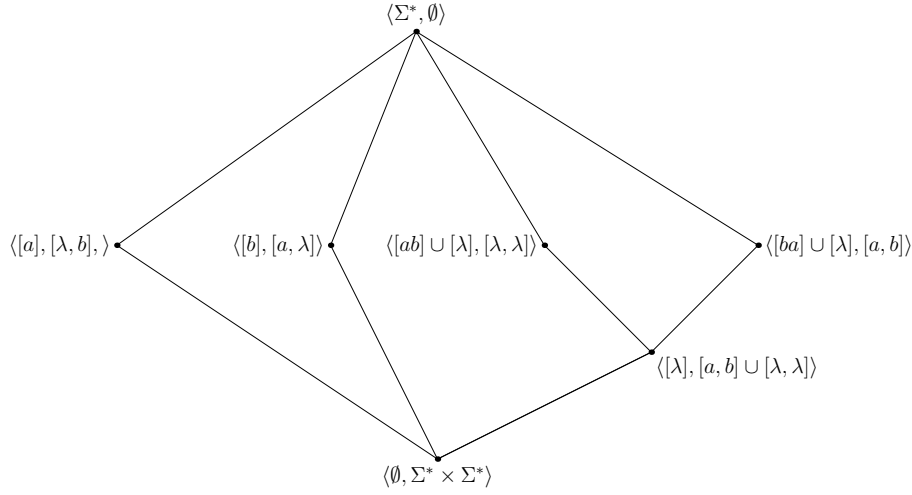


Fig. 2 The syntactic concept lattice of the regular language $(ab)^*$.

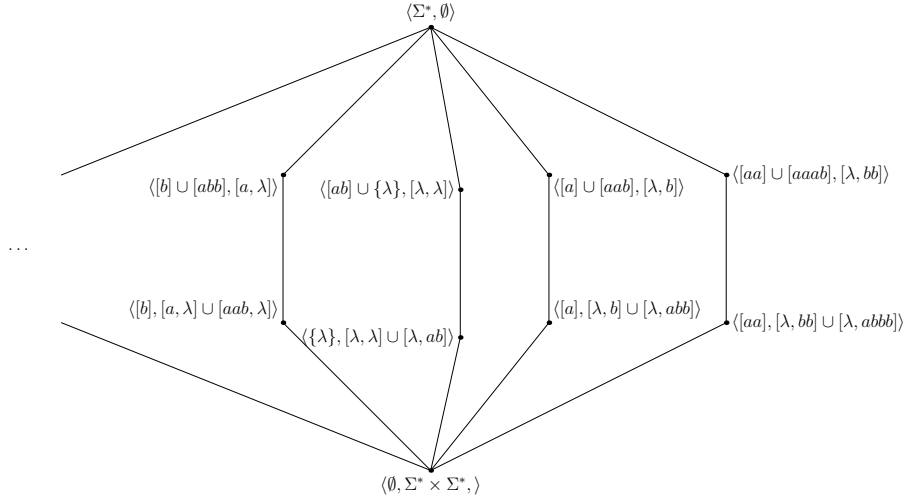


Fig. 3 A fragment of the syntactic concept lattice of the context-free language $\{a^n b^n \mid n \geq 0\}$. The lattice is finite so we just show the components generated by the short strings. The remainder of the lattice is structurally similar to the fragment shown here and contains concept containing strings like a^n, b^n and so on, where $n > 2$.

Proof Let $O_1^c = \{w \in \{a, b\}^* \mid |w|_a \neq |w|_b\}$. Here $|w|_a$ means the number of occurrences in the string w of the symbol a . Define $c(w) = |w|_a - |w|_b$, and $c(l, r) = -c(lr)$. We can see that $u \equiv_L v$ iff $c(u) = c(v)$. For each $X \subseteq \mathbb{Z}$, define the concept $C(X) = \langle \{w \mid c(w) \in X\}, \{(l, r) \mid c(l, r) \notin X\} \rangle$. It is easy to verify that these are concepts, and therefore $\mathfrak{B}(L)$ is in bijection with $2^{\mathbb{Z}}$.

Note that in this example, the syntactic monoid of O_1^c is isomorphic to \mathbb{Z} under addition.⁵

3.1 Concatenation

We now define the natural concatenation operation and its associated residuals.

Definition 19 For two closed sets of strings R, T define $R \circ T = (RT)^{\triangleright\triangleleft}$. For two concepts, $\langle S_R, C_R \rangle \circ \langle S_T, C_T \rangle = \langle (S_R S_T)^{\triangleright\triangleleft}, (S_R S_T)^{\triangleright} \rangle = \mathcal{C}(S_R S_T)$.

Definition 20 Given a language L , we define the unit in the lattice to be $\lambda_L = \langle \{\lambda\}^{\triangleright\triangleleft}, \{\lambda\}^{\triangleright} \rangle$.

The closure operation interacts nicely with the concatenation operation. Just as distributional equivalence (\equiv_L) is a congruence of the monoid, implying that $[uv] \supseteq [u][v]$, so we have a related result for the lattice operations, lifted to sets of strings. Indeed as we shall see later, this defines a congruence on sets of strings – a congruence on $\mathcal{P}(\Sigma^*)$.

Proposition 27 Suppose we have two sets of strings Y, Z . Then

$$(YZ)^{\triangleright\triangleleft} \supseteq Y^{\triangleright\triangleleft} Z^{\triangleright\triangleleft}$$

.

Proof [GJKO07, Lei12] Suppose $y \in Y^{\triangleright\triangleleft}$ and $z \in Z^{\triangleright\triangleleft}$. So $\{y\}^{\triangleright} \supseteq Y^{\triangleright}$ and $\{z\}^{\triangleright} \supseteq Z^{\triangleright}$. Take $(l, r) \in (YZ)^{\triangleright}$. So $lYZr \subseteq L$. This means that $(l, Zr) \subseteq Y^{\triangleright}$. So $(l, Zr) \subseteq Y^{\triangleright} \subseteq \{y\}^{\triangleright}$. So $lyZr \subseteq L$. This means that $(ly, r) \subseteq Z^{\triangleright} \subseteq \{z\}^{\triangleright}$. So $lyzr \in L$. So $(l, r) \in (yz)^{\triangleright}$ and so $(l, r) \in (Y^{\triangleright\triangleleft} Z^{\triangleright\triangleleft})^{\triangleright}$. Therefore $(YZ)^{\triangleright} \subseteq (Y^{\triangleright\triangleleft} Z^{\triangleright\triangleleft})^{\triangleright}$. Therefore $(YZ)^{\triangleright\triangleleft} \supseteq (Y^{\triangleright\triangleleft} Z^{\triangleright\triangleleft})^{\triangleright\triangleleft} \supseteq Y^{\triangleright\triangleleft} Z^{\triangleright\triangleleft}$.

If $Y = \{u\}$ and $Z = \{v\}$ then this means that $\{uv\}^{\triangleright\triangleleft} \supseteq \{u\}^{\triangleright\triangleleft} \{v\}^{\triangleright\triangleleft}$; contrast this with the result in Proposition 4.

Proposition 28 Suppose we have sets of strings X, Y, Z such that $X \supseteq YZ$. Then $X^{\triangleright\triangleleft} \supseteq Y^{\triangleright\triangleleft} Z^{\triangleright\triangleleft}$.

Proposition 29 For any sets of strings X, Y , $(XY)^{\triangleright\triangleleft} = (X^{\triangleright\triangleleft} Y^{\triangleright\triangleleft})^{\triangleright\triangleleft}$.

Proof By Proposition 27, $XY^{\triangleright\triangleleft} \supseteq X^{\triangleright\triangleleft} Y^{\triangleright\triangleleft}$. So $XY^{\triangleright\triangleleft} = (XY^{\triangleright\triangleleft})^{\triangleright\triangleleft} \supseteq (X^{\triangleright\triangleleft} Y^{\triangleright\triangleleft})^{\triangleright\triangleleft}$. But $XY \subseteq X^{\triangleright\triangleleft} Y^{\triangleright\triangleleft}$, so $XY^{\triangleright\triangleleft} \subseteq (X^{\triangleright\triangleleft} Y^{\triangleright\triangleleft})^{\triangleright\triangleleft}$.

Proposition 30 $\langle \mathfrak{B}(L), \circ, \lambda_L \rangle$ is a monoid.

⁵ Note that this implies the existence of an interesting language class: the class of languages with a countable concept lattice.

Proof We can verify that the operation \circ is associative easily using the previous lemmas. Given three closed sets of strings X, Y, Z , we can see that

$$((XY)^{\triangleright\triangleleft}Z)^{\triangleright\triangleleft} = (XYZ)^{\triangleright\triangleleft} = (X(YZ)^{\triangleright\triangleleft})^{\triangleright\triangleleft}$$

Similarly since $X = X\{\lambda\}$,

$$X^{\triangleright\triangleleft} = (X\{\lambda\}^{\triangleright\triangleleft})^{\triangleright\triangleleft} = (\{\lambda\}^{\triangleright\triangleleft}X)^{\triangleright\triangleleft}$$

which establishes that λ_L is the identity.

Proposition 31 $\mathfrak{B}(L)$ is a complete idempotent semiring.

Proof We define the bottom element as $\perp = \mathcal{C}(\emptyset)$; \circ and \vee are defined as before. In order to verify that it is a complete idempotent semiring, we need to check the axioms. By Proposition 30, we have the relevant monoid structure. By the definition of \vee it is associative, commutative and idempotent. If X is closed then by Proposition 20, $X = (X \cup \emptyset^{\triangleright\triangleleft})^{\triangleright\triangleleft}$ so \perp is a unit with respect to \vee . The only remaining point to check that $X \circ \bigvee_i Y_i = \bigvee_i (X \circ Y_i)$. Suppose R, S_i are closed sets of strings; then $R(\bigcup_i S_i) = \bigcup_i RS_i$.

$$(R(\bigcup_i S_i))^{\triangleright\triangleleft} = (R(\bigcup_i S_i))^{\triangleright\triangleleft} = (\bigcup_i RS_i)^{\triangleright\triangleleft} = (\bigcup_i (RS_i)^{\triangleright\triangleleft})^{\triangleright\triangleleft}.$$

Proposition 32 By Proposition 2, $\mathfrak{B}(L)$ is a residuated lattice.

We can now give a complete definition of all of the operations in this residuated lattice using only closed sets of strings.

Definition 21 [Lei12] The structure consisting of $\{X \subseteq \Sigma^* \mid X = X^{\triangleright\triangleleft}\}$, with constants $\perp = \emptyset^{\triangleright\triangleleft}$, $\top = \Sigma^*$, $1 = \lambda^{\triangleright\triangleleft}$ and partial order $X \leq Y$ iff $X \subseteq Y$, operations $X \vee Y = (X \cup Y)^{\triangleright\triangleleft}$, $X \wedge Y = X \cap Y$, and $X \circ Y = (XY)^{\triangleright\triangleleft}$, $X/Y = \{z \mid zY \subseteq X\}$, $X \backslash Y = \{z \mid Xz \subseteq Y\}$ is a residuated lattice.

In what follows we will largely take the concepts to be just closed sets of strings, implicitly using the mapping $\langle S, C \rangle \rightarrow S$ where appropriate.

4 Universal property

Some rhetoric first – we want a system of syntactic categories to interpret our grammars [GR62]. We have at a minimum two operations – concatenation and union – if we restrict ourselves to CFGs. One such system might be $\mathcal{P}(\Sigma^*)$, but that is in a sense too complex. For each language, we want the simplest such system: an algebraic structure that contains enough categories to represent the nonterminals but no more. In this section we will show that for any language there is a single simplest system of categories, unique up to isomorphism, which is as we shall see the syntactic concept lattice; we provide a formal statement of this in Proposition 51.

Definition 22 Given a CIS, S , we say that an equivalence relation \equiv on the elements of S is a CIS-congruence if for all $X, Y, Z \in S$:

- $X \equiv Y$ implies $X \circ Z \equiv Y \circ Z$ and $Z \circ X \equiv Z \circ Y$
- $X \equiv Y$ implies $X \vee Z \equiv Y \vee Z$
- For any collection of sets X_i, Y_i such that $X_i \equiv Y_i$, $\bigvee_i X_i \equiv \bigvee_i Y_i$.

Definition 23 Given two CIS, A and B a function $h : A \rightarrow B$ is a CIS-morphism iff

- $h(x \vee y) = h(x) \vee h(y)$
- For any collection x_i , $h(\bigvee_i x_i) = \bigvee_i h(x_i)$
- $h(x \circ y) = h(x) \circ h(y)$.

Proposition 33 If h is a surjective CIS-morphism from A to B , then $h(1_A) = 1_B$ and $h(\perp_A) = \perp_B$.

Proof $h(a) = h(1_A \circ a) = h(1_A) \circ h(a)$ and $h(a) = h(a \circ 1_A) = h(a) \circ h(1_A)$. So $1_B = h(1_A) \circ 1_B = h(1_A)$. $h(a) = h(a \vee \perp_A) = h(a) \vee h(\perp_A)$. So $h(\perp_A) = h(\perp_A) \vee \perp_B = \perp_B$.

Proposition 34 Suppose $x \leq y$; then $h(x) \leq h(y)$.

Proof Note that $x \leq y$ iff $x = x \vee y$. If this is true then $h(x) = h(x \vee y) = h(x) \vee h(y)$ implies $h(x) \leq h(y)$.

We now define the residual map; informally this is a map which is analogous to the inverse.

Definition 24 If we have a CIS-morphism h , from CIS A to B , we define h^* to be the map from B to A defined as $h^*(y) = \bigvee \{x \in A : h(x) \leq y\}$.

Now h and h^* form what is called a residuated pair.

Proposition 35 For all $x \in A, y \in B$, $h(x) \leq y$ iff $x \leq h^*(y)$

Proof Suppose $h(x) \leq y$. $h^*(y) = \bigvee_{z \in A: h(z) \leq y} z$. But since $h(x) \leq y$ this means that one of these z will be x , so $h^*(y) \geq x$. Conversely suppose $x \leq h^*(y)$; Then $h(x) \leq h(h^*(y)) = h(\bigvee_{z \in A: h(z) \leq y} z) = \bigvee_{z \in A: h(z) \leq y} h(z) \leq y$. Therefore $h(x) \leq y$.

Proposition 36 $h(h^*(y)) \leq y$ and $h^*(h(x)) \geq x$

Proof $h^*(y) \leq h^*(y)$ so $h(h^*(y)) \leq y$. $h(x) \leq h(x)$ so $x \leq h^*(h(x))$.

Proposition 37 Both h and h^* are monotonic.

Note that if A is $\mathcal{P}(\Sigma^*)$ then the homomorphism is uniquely defined just by the values of the elements of Σ ; once we know what $h(\{a\})$ is for each element $a \in \Sigma$ then this fixes the value for every string and thus every set of strings. This is rather like the way a linear map (a homomorphism of vector spaces) is fixed by the value of the function at the elements of a basis.

We are concerned exclusively with homomorphisms that arise from congruences of $\mathcal{P}(\Sigma^*)$ and are therefore all surjective. In this case we can slightly strengthen the results.

Proposition 38 *If h is a surjective CIS-homomorphism then $h(h^*(y)) = y$.*

Proof By definition $h^*(y) = \bigvee \{x \in A : h(x) \leq y\}$. Consider $\{x \in A : h(x) = y\}$. Since h is surjective, this is nonempty. Therefore $\bigvee \{x \in A : h(x) \leq y\} = \bigvee \{x \in A : h(x) = y\}$ by monotonicity of h . So $h(h^*(y)) = h(\bigvee \{x \in A : h(x) = y\}) = \bigvee \{y\} = y$.

Proposition 39 *If $h : A \rightarrow B$ is a surjective CIS-homomorphism then for any $X, Y \in B$, $h^*(X \circ Y) \geq h^*(X) \circ h^*(Y)$.*

Proof Consider $h(h^*(X) \circ h^*(Y)) = h(h^*(X)) \circ h(h^*(Y)) = X \circ Y$. So $h^*(X) \circ h^*(Y) \in \{z \in A \mid h(z) \leq X \circ Y\}$. Now $h^*(X \circ Y) = \bigvee \{z \in A \mid h(z) \leq X \circ Y\} \geq h^*(X) \circ h^*(Y)$.

We now define the idea of a CIS recognising a language which differs in some respects to the earlier definition of a monoid recognising a language.

Definition 25 We say that a CIS B recognizes L through the morphism h if h is a surjective morphism from $\mathcal{P}(\Sigma^*) \rightarrow B$ such that $h^*(h(L)) = L$.

Example 7 $\mathcal{P}(\Sigma^*)$ recognizes L through the identity morphism.

Recall that this morphism is fixed just by the elements of $h(a)$ for $a \in \Sigma$. So if we have a morphism from $\mathcal{P}(\Sigma^*) \rightarrow B$ then $h(L)$ is the unique element that might represent L . Therefore, $h^*(h(L)) \geq L$ by the previous proposition. The condition thus just states that this element does in fact correspond to L and not some other larger set of strings. For example if B is the single element CIS, then $h^*(h(L)) = \Sigma^*$.

Having established these general properties of CIS-morphisms, and congruences we can now consider the specific examples we are interested in.

Definition 26 [Yos11] Given a language L we define an equivalence relation on sets of strings i.e. on $\mathcal{P}(\Sigma^*)$, where the set of strings X is congruent to the set of strings Y , written $X \cong_L Y$ iff $X^\triangleright = Y^\triangleright$.

This congruence is on sets of strings – the syntactic congruence is a congruence of strings. They coincide on singleton sets of strings.

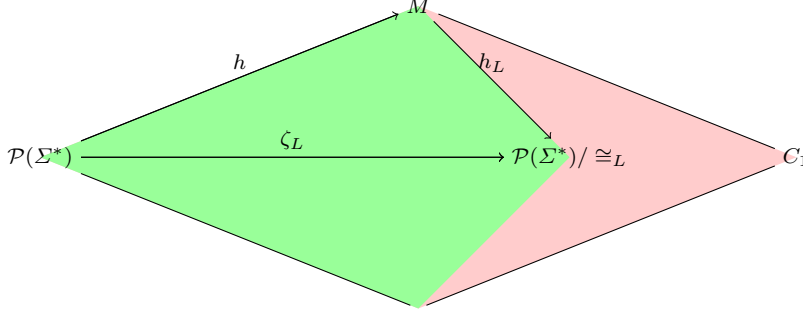
Proposition 40 *The relation \cong_L is a CIS-congruence.*

Proof For all sets of strings X, Y, W, Z

- If $X^\triangleright = Y^\triangleright$ and $W^\triangleright = Z^\triangleright$ then by Proposition 29, $XW^{\triangleright\triangleleft} = (X^{\triangleright\triangleleft}W^{\triangleright\triangleleft})^{\triangleright\triangleleft} = (Y^{\triangleright\triangleleft}Z^{\triangleright\triangleleft})^{\triangleright\triangleleft} = YZ^{\triangleright\triangleleft}$. So $(XW)^\triangleright = (YZ)^\triangleright$.
- If $X^\triangleright = Y^\triangleright$ and $W^\triangleright = Z^\triangleright$ then $(X \cup W)^\triangleright = X^\triangleright \cap W^\triangleright = Y^\triangleright \cap Z^\triangleright = (Y \cup Z)^\triangleright$. (or by Proposition 20)
- Suppose X_i, Y_i are families of sets of strings indexed by I such that $X_i \cong_L Y_i$ for all $i \in I$. Then $(\bigvee_i X_i)^\triangleright = \bigcap_i X_i^\triangleright = \bigcap_i Y_i^\triangleright = (\bigvee_i Y_i)^\triangleright$.

Proposition 41 *For any language L , $\mathfrak{B}(L)$ is CIS-isomorphic to $\mathcal{P}(\Sigma^*) / \cong_L$.*

Fig. 4 The syntactic concept lattice: the coarsest congruence of $\mathcal{P}(\Sigma^*)$ that recognises L . Fine congruences are on the left and on the far right is the coarsest congruence of all with only one congruence class. Congruences that recognise L are shaded green. Congruences that do not are shaded red. C_1 is the trivial one element CIS. If M is a CIS that recognises L through the morphism h then there is a homomorphism from M to $\mathcal{P}(\Sigma^*)/\cong_L$, h_L such that $\zeta_L = h \cdot h_L$.



Proof We define the natural function $\pi : \mathfrak{B}(L) \rightarrow \mathcal{P}(\Sigma^*)/\cong_L$, given by $\pi(X) = [X]_{\cong_L}$. We can verify that this is a bijection and that the operations of \circ and \vee are the same.

So the SCL is the quotient of $\mathcal{P}(\Sigma^*)$ under the congruence $X \cong Y$ iff $X^{\triangleright\triangleleft} = Y^{\triangleright\triangleleft}$ iff $X^{\triangleright} = Y^{\triangleright}$. The relation to the syntactic congruence thus becomes even clearer: the syntactic monoid is the quotient of Σ^* under the congruence $u \equiv v$ iff $\{u\}^{\triangleright} = \{v\}^{\triangleright}$.

There is a potential confusion here between sets of sets of strings and sets of strings. The relation \cong_L is a relation between sets of strings: $X \cong_L Y$ where X, Y are sets of strings. The equivalence class $[X]_{\cong_L}$ is thus a set of sets of strings; a collection whose elements are sets of strings that are congruent to X . Now it so happens that if Y, Z are two sets of strings in $[X]_{\cong_L}$ then so is $Y \cup Z$ (since it is a congruence wrt \vee), as is indeed the union of all the elements of $[X]_{\cong_L}$.

Proposition 42 *Let $[X]_{\cong_L}$ be the set of all sets of strings congruent to X . Then $\bigcup\{Y \mid Y \cong_L X\} = X^{\triangleright\triangleleft} \cong_L X$. Or, $\bigcup\{Y \mid Y \cong_L X\} \in [X]_{\cong_L}$.*

$[X]_{\cong_L}$ is a set of sets of strings. The union of all the sets in $[X]_{\cong_L}$ is a set of strings $\bigcup_{Y \in [X]} Y$ which is equal to $X^{\triangleright\triangleleft}$ which happens to be in $[X]$. So \cong is a partition of the set of all sets of strings. A given set of strings will be in only one concept – its closure. A given string u may be in a variety of closed sets of strings, some large and some small – the largest will be Σ^* and the smallest will be $\{u\}^{\triangleright\triangleleft}$.

We can now start to state the universal property of the syntactic concept lattice.

Proposition 43 *The natural map $\zeta_L : \mathcal{P}(\Sigma^*) \rightarrow \mathfrak{B}(L)$ given by $\zeta(X) = X^{\triangleright\triangleleft}$ is a CIS-homomorphism.*

Proposition 44 *The residual of this is $\zeta_L^* : \mathfrak{B}(L) \rightarrow \mathcal{P}(\Sigma^*)$ which is just the identity on sets of strings; the natural embedding of the closed sets of strings in $\mathcal{P}(\Sigma^*)$; $\zeta_L^*(\langle S, C \rangle) = S$.*

We will now show that this is the coarsest congruence that recognises L . If it is any coarser (i.e fewer larger classes) then it will not have L in it. Therefore this is the unique smallest structure that we can use to interpret this language.

Proposition 45 $\mathfrak{B}(L)$ recognizes L , through the morphism $X \rightarrow X^{\triangleright\triangleleft}$.

Proof Immediate; since $L^{\triangleright\triangleleft} = L$, so $h(L) = L$ and therefore $h^*(h(L)) = L$.

We now prove an important technical lemma; informally it says that any lattice that recognizes this must be 'finer' (larger, more elaborate) than the syntactic context lattice.

Lemma 1 *Suppose, A is some CIS that recognizes L through a morphism h . Suppose there are two sets of strings X, Y such that $h(X) = h(Y)$; then $\zeta_L(X) = \zeta_L(Y)$. (i.e. $X^{\triangleright} = Y^{\triangleright}$)*

Proof Suppose $(l, r) \in X^{\triangleright}$. That means that $lXr \subseteq L$; so $h(lXr) \leq h(L)$ since h is monotonic. This implies that $h(\{l\}) \circ h(X) \circ h(\{r\}) \leq h(L)$ using the fact that h is a homomorphism. So $h(\{l\}) \circ h(Y) \circ h(\{r\}) \leq h(L)$ since $h(X) = h(Y)$. Therefore $h(\{l\}Y\{r\}) \leq h(L)$ since it is a homomorphism. So $h^*(h(\{l\}Y\{r\})) \leq h^*(h(L))$ since h^* is monotonic. Now $h^*(h(L)) = L$ since A recognises L . So using the fact that $h^*(h(X)) \geq X$ we have $\{l\}Y\{r\} \leq h^*(h(\{l\}Y\{r\})) \leq h^*(h(L)) = L$. So $(l, r) \in Y^{\triangleright}$. The converse argument is identical and so $X^{\triangleright} = Y^{\triangleright}$.

See Figure 4 for a diagram that illustrates this. The following proposition says that the SCL is (isomorphic to) the smallest CIS that recognizes L ; it is the terminal element of the appropriate category.

Proposition 46 *If we have a lattice B that recognizes L through a morphism h , then there is a morphism h_L from B to $\mathfrak{B}(L)$ such that $\zeta_L = h \circ h_L$. Therefore the smallest CIS that recognises L is unique up to isomorphism.*

Proof Define $h_L(h(X)) = \zeta_L(X)$. This is well defined by Lemma 1. We can verify easily that it is a morphism.

This is again a consequence of the isomorphism theorems.

5 CFG Morphisms

We can now make clear the relation between these morphisms between semirings, and representations such as CFGs. The semirings, on our view, are merely a way of talking about CFGs and their morphisms in a more algebraic way;

a method to talk about the ‘semantics’ of CFGS.⁶ We now make the link to CFGS; and in particular we will link the semantic notion of a morphism – a homomorphism between complete idempotent semirings – with the syntactic notion – a morphism between grammars.

We start by defining some standard notions of mappings between CFGS; the definitions are straightforward extensions of the idea of morphisms between finite automata [Lom07]; we will give them in full since we have not so far been able to find them in the literature.

Definition 27 Assume we have two CFGS, $G_1 = \langle \Sigma, V_1, S_1, P_1 \rangle$ and $G_2 = \langle \Sigma, V_2, S_2, P_2 \rangle$, defined over the same terminal alphabet Σ . G_1 is a subgrammar of G_2 iff $V_1 \subseteq V_2$, $P_1 \subseteq P_2$ and $S_1 = S_2$.

Proposition 47 If G_1 is a subgrammar of G_2 , then $L(G_1) \subseteq L(G_2)$.

Definition 28 Assume we have two CFGS, $G_1 = \langle \Sigma, V_1, S_1, P_1 \rangle$ and $G_2 = \langle \Sigma, V_2, S_2, P_2 \rangle$, defined over the same terminal alphabet Σ . Let ϕ be a function from V_1 to V_2 ; we extend it to a function from $V_1 \cup \Sigma$ to $V_2 \cup \Sigma$ by setting $\phi(a) = a$ for all $a \in \Sigma$, and thence to a function from $(V_1 \cup \Sigma)^*$ by saying that $\phi(\alpha\beta) = \phi(\alpha)\phi(\beta)$ and $\phi(\lambda) = \lambda$. For a production $N \rightarrow \alpha$ we say that $\phi(N \rightarrow \alpha) = \phi(N) \rightarrow \phi(\alpha)$. A *morphism* ϕ from G_1 to G_2 is such a function from V_1 to V_2 , such that $\phi(S_1) = \phi(S_2)$, and $\phi(P_1) \subseteq P_2$.

Example 8 Suppose G_1 has the three nonterminals S, A, B and productions $S \rightarrow AB, S \rightarrow ASB$ and $A \rightarrow a, B \rightarrow b$. $\mathcal{L}(G_1) = \{a^n b^n | n > 0\}$. Suppose G_2 has two nonterminals S, X and productions $S \rightarrow XX, S \rightarrow XSX$ and $X \rightarrow a, X \rightarrow b$. Then ϕ defined as $\phi(A) = X, \phi(B) = X, \phi(S) = S$ is a morphism. $\phi(A \rightarrow a) = X \rightarrow a$, $\phi(B \rightarrow b) = X \rightarrow b$, and so on.

Proposition 48 If there is a morphism from G_1 to G_2 then $L(G_1) \subseteq L(G_2)$.

We can state a more general result as follows.

Proposition 49 If there is a morphism from G_1 to G_2 then for all nonterminals N in G_1 :

- $\mathcal{L}(G_1, N) \subseteq \mathcal{L}(G_2, \phi(N))$
- $\mathcal{C}(G_1, N) \subseteq \mathcal{C}(G_2, \phi(N))$

Proof Take a derivation step $\beta N \gamma \Rightarrow_{G_1} \beta \alpha \gamma$ for some production $N \rightarrow \alpha \in P_1$. Then, since there is a morphism, ϕ , there is a production $\phi(N) \rightarrow \phi(\alpha)$ in P_2 and thus $\phi(\beta)\phi(N)\phi(\gamma) \Rightarrow_{G_2} \phi(\beta)\phi(\alpha)\phi(\gamma)$.

Suppose $N \Rightarrow \alpha_1 \Rightarrow \dots \alpha_n \Rightarrow u$ is a derivation in G_1 . Then $\phi(N) \Rightarrow \phi(\alpha_1) \Rightarrow \dots \phi(\alpha_n) \Rightarrow \phi(u) = u$ is a derivation of u from $\phi(N)$ wrt G_2 .

Suppose $S \Rightarrow \alpha_1 \Rightarrow \dots \alpha_n \Rightarrow lNr$ is a derivation in G_1 . Then $\phi(S) \Rightarrow \phi(\alpha_1) \Rightarrow \dots \phi(\alpha_n) \Rightarrow \phi(lNr) = l\phi(N)r$ is a derivation in G_2 .

⁶ In this context the syntax/semantics distinction refers to the distinction between the rules of the grammar, and the language it defines, and has nothing to do with the meaning of a sentence or utterance [PS84].

Note here that there is not the usual duality between contexts and substrings here – rather both sets get bigger (or at least no smaller) when we go from G_1 to G_2 .

Corollary 1 *If there is a morphism from G_1 to G_2 then for all nonterminals N in G_2 :*

- $\bigcup_{M:\phi(M)=N} \mathcal{L}(G_1, M) \subseteq \mathcal{L}(G_2, N)$
- $\bigcup_{M:\phi(M)=N} \mathcal{C}(G_1, M) \subseteq \mathcal{C}(G_2, N)$

Definition 29 The image of G_1 , $\phi(G_1) = \langle \Sigma, \phi(V_1), S_2, \phi(P_1) \rangle$ is a CFG, also which is called the morphic image of G_1 under the morphism ϕ . This will be a subgrammar of G_2 .

Definition 30 Given a CFG $G = \langle \Sigma, V, S, P \rangle$, and a function ψ from V into some arbitrary set X , we can define the morphic image of G to be the grammar $\psi(G) = \langle \Sigma, \psi(V), \psi(S), \psi(P) \rangle$. Clearly ψ is a morphism from G to $\psi(G)$.

A brief discussion: if ψ is injective – i.e. it is not the case that two distinct nonterminals are mapped to the same symbol, then this is just a relabeling of G and it defines the same language. The more interesting case is where we have two or more nonterminals that are mapped to the same symbol: two distinct nonterminals M, N such that $\psi(M) = \psi(N)$. The new grammar then will be strictly smaller than the original grammar. In this case it may be that the new grammar will define a larger language. To take an extreme case, we can map any grammar in CNF to a grammar with one nonterminal which will normally define the language Σ^* or Σ^+ . This again is not very interesting – what we are most concerned with is the case when the resulting grammar, though smaller, defines the same language.

If there is a bijective morphism between two grammars, then we say they are isomorphic – they are identical up to a relabeling of nonterminals. We now consider the case where the morphic image of the grammar defines the same language.

Definition 31 A morphism from G_1 to G_2 is called *exact* if $L(G_1) = L(G_2)$.

Definition 32 For a CFG G_1 if there is a grammar G_2 such that there is an exact morphism $\phi : G_1 \rightarrow G_2$, (i.e. such that $L(G_1) = L(G_2)$), and there are two nonterminals M, N such that $\phi(M) = \phi(N)$, then we say that M and N are mergeable.

Definition 33 We say that a CFG is *minimal* if it does not have any mergeable nonterminals.

Note that this definition does not mean that this is the globally smallest grammar for the language $L(G)$. It might even have some redundant states that could be removed. Clearly if a grammar has two nonterminals that are mergeable then we can merge them and get another grammar which defines the same language but is strictly smaller. Suppose we are interested in minimal CFGs; then we want to find CFGs that do not have mergeable states; because if we do have mergeable states then the grammar is not minimal.

6 The universal morphism

We can now join up the (syntactic) CFG-morphisms and the (semantic) CIS-homomorphisms through the following lemma; any CIS that recognizes a language induces a morphism of the grammar where we map two nonterminals together if they 'mean' the same thing. If the CIS recognises L then we will show that the morphism is exact.

Lemma 2 *Suppose B is a CIS and h a surjective CIS homomorphism from $\mathcal{P}(\Sigma^*) \rightarrow B$. Then let G be a context-free grammar. Let ϕ be the CFG-morphism given by $\phi(N) = h(\mathcal{L}(G, N))$. Then for all nonterminals N in G ,*

$$\mathcal{L}(\phi(G), \phi(N)) \subseteq h^*(\phi(N))$$

Proof We will assume for clarity of exposition that the grammar is in CNF – the proof of the more general case is straightforward but less legible.

Suppose we have a rule $N \rightarrow PQ$ in G . Therefore $\mathcal{L}(G, N) \supseteq \mathcal{L}(G, P)\mathcal{L}(G, Q)$. So $h(\mathcal{L}(G, N)) \supseteq h(\mathcal{L}(G, P)) \circ h(\mathcal{L}(G, Q))$.

Then $h^*(h(\mathcal{L}(G, N))) \supseteq h^*(h(\mathcal{L}(G, P)) \circ h(\mathcal{L}(G, Q)))$ by monotonicity of h^* . By Proposition 39 $h^*(h(\mathcal{L}(G, N))) \supseteq h^*(h(\mathcal{L}(G, P))) \circ h^*(h(\mathcal{L}(G, Q)))$. Similarly suppose we have a rule $N \rightarrow a$ in G . So $h(\mathcal{L}(G, N)) \supseteq h(\{a\})$, and $h^*(h(\mathcal{L}(G, N))) \supseteq h^*(h(\{a\}))$.

Now suppose that $N' \xrightarrow{*}_{\phi(G)} w$; we want to show that $w \in h^*(N')$. We proceed by induction on the length of the derivation. Base case: the production is of length 1. Suppose $N' \rightarrow a$ is a production in $\phi(G)$. Then for any production $N \rightarrow a$ in G such that $N' = \phi(N)$, $h^*(h(\mathcal{L}(G, N))) \supseteq h^*(h(\{a\})) \supseteq \{a\}$. Suppose it is true for all derivations of length at most k , and let $N' \xrightarrow{*}_{\phi(G)} w$ be a derivation of length $k+1$ that starts with $N' \rightarrow P'Q' \rightarrow uv = w$, where $P' \xrightarrow{*}_{\phi(G)} u$ and $Q' \xrightarrow{*}_{\phi(G)} v$. these last two derivations must be of length at most k .

Then consider any production $N \rightarrow PQ$ in G such that $\phi(N) = N'$, $\phi(P) = P'$ and $\phi(Q) = Q'$ (of which there must be at least one). Now by the inductive hypothesis, $u \in h^*(P')$ and $v \in h^*(Q')$. So $uv \in h^*(P')h^*(Q') = h^*(h(\mathcal{L}(G, P))) \circ h^*(h(\mathcal{L}(G, Q)))$. Now since $h^*(h(\mathcal{L}(G, N))) \supseteq h^*(h(\mathcal{L}(G, P))) \circ h^*(h(\mathcal{L}(G, Q)))$ this means that $uv \in h^*(h(\mathcal{L}(G, N)))$. So $w \in h^*(\phi(N))$. By induction, it is therefore true for derivations of any length.

Clearly this implies the obvious fact that if h is the identity (and therefore so is h^*) this gives an exact morphism; the morphism which merges nonterminals which generate the same sets of strings.

As a simple corollary we have now the following lemma, which explains our definition of a CIS recognising a language through a morphism.

Proposition 50 *Let G be a CFG that defines the language L and B a CIS that recognises L through the morphism h . Then the morphism induced by map $\phi : N \rightarrow h(\mathcal{L}(G, N))$ is an exact CFG-morphism from G to $\phi(G)$.*

Proof Since it is a morphism we know that $\mathcal{L}(\phi(G)) \supseteq \mathcal{L}(G)$. Since B recognises L , $h^*(h(L)) = L$, so $\mathcal{L}(\phi(G), \phi(S)) \subseteq h^*(\phi(S)) = L$.

Note that this is only an implication: we can have a homomorphism that does not recognise the language, but where the induced grammar morphism is still exact, i.e. defines the same language. This could occur for example when we merge two nonterminals M, N where $\mathcal{C}(G, M) \odot \mathcal{L}(G, N) \subseteq L$ and $\mathcal{C}(G, N) \odot \mathcal{L}(G, M) \subseteq L$.

We can now define the universal morphism.

Definition 34 Suppose G is a grammar that defines a language L ; the *universal morphism* ϕ_L is the CFG-morphism that for any nonterminal N of G maps $N \rightarrow \mathcal{L}(G, N)^{\triangleright\triangleleft}$.

Note that the universal morphism is an exact morphism by the previous lemma.

We can now state the following proposition, which informally states that if a grammar is minimal then there is at most one nonterminal for each concept.

Proposition 51 *if the grammar is minimal, then the universal morphism is injective.*

Proof If there were two nonterminals for a concept, the image under the universal morphism would have one fewer nonterminal and would thus be smaller than the original grammar, contradicting the assumption of minimality.

We may not be able to compute, for an arbitrary CFG whether two nonterminals are mergeable, but that is to a certain extent irrelevant. This proposition tells us that we can restrict ourselves to nonterminals that are concepts. We have two consequences – one is that that for every context free language we have a grammar where the nonterminals are closed sets of strings; secondly the simplest grammars for every language will be of this type. In the next section we will discuss the special case of regular languages, where the exact computations are possible.

7 Regular languages

In the case of regular languages, we know the lattice will be finite and this simplifies the analysis in a number of respects. In this case, all of the concepts can be defined using a finite number of substrings. Accordingly, it is possible to define the same structure using only idempotent semirings rather than complete idempotent semirings. In this case the construction is called the *syntactic semiring* and was introduced in [Pol01, Pol03]. We briefly summarize the syntactic semiring in a notation consistent with this paper. Define $\mathcal{F}(X)$ to be the set of all finite subsets of a set X . Then $\mathcal{F}(\Sigma^*)$ is the free idempotent semiring over Σ . Consider a language $L \subseteq \Sigma^*$. We can define a congruence on $\mathcal{F}(\Sigma^*)$, given X, Y which are elements of $\mathcal{F}(\Sigma^*)$, $X \sim_L Y$ iff $X^\triangleright = Y^\triangleright$. That

is to say merely the restriction of the congruence to the collection of finite sets of strings. The quotient $\mathcal{F}(\Sigma^*)/\sim_L$ is the syntactic semiring. It can be shown that this is isomorphic to the SCL in the case of regular languages. Note that since $\mathcal{F}(\Sigma^*)$ is countable so is the syntactic semiring. Consider the example O_1^c . In this case the syntactic semiring is countable, whereas the syntactic concept lattice is uncountable; and there is no element that corresponds to L .

We can also construct a canonical representation. This is the universal CFG for the language and will contain a morphic image of every (binarised) context free grammar for the language. We emphasize that in the case of a non-regular context-free language, we can do the same construction but the resulting object will be infinite and thus not strictly speaking a grammar. Nonetheless, it will contain as with the finite case an image of every grammar for the language.

Definition 35 Given a regular language L , the universal context free grammar for the language $\mathfrak{G}(L)$ is the CFG $\langle V, \mathcal{C}(L), \Sigma, P \rangle$, where V is the set of concepts of L , and P is the set of productions $\{Q \rightarrow RS \mid Q \geq R \circ S\} \cup \{N \rightarrow a \mid a \in N, a \in \Sigma \cup \{\lambda\}\}$

We can now demonstrate an exact correspondence between the derivations in the grammar, and the pair of sets of contexts and sets of strings that constitute a nonterminal in the grammar.

Proposition 52 *For any language L and any concept $X = (S, C)$, we have*

- $\mathcal{L}(\mathfrak{G}(L), (S, C)) = S$
- $\mathcal{C}(\mathfrak{G}(L), (S, C)) = C$

Proof By induction on the length of a derivation. Suppose $S \xrightarrow{*} lXr$. We know $\mathcal{C}(l) \xrightarrow{*} l$ and $\mathcal{C}(r) \xrightarrow{*} r$, by part 1, and $S \geq \mathcal{C}(l) \circ X \circ \mathcal{C}(r)$ which gives us the two rules.

Proposition 53 *Suppose we have a regular language L and a CFG G such that $\mathcal{L}(G) = L$. Then there is an exact morphism from G to $\mathfrak{G}(L)$.*

Proof The morphism $N \rightarrow (\mathcal{L}(G, N)^{\triangleright^L}, \mathcal{L}(G, N)^{\triangleright})$ is such a morphism.

Proposition 54 *The universal grammar does not have any mergeable states.*

Proposition 55 *Given a finite automaton generating a language L , we can effectively construct $\mathfrak{G}(L)$.*

We sketch the technical details for constructing a structure isomorphic to $\mathfrak{B}(L)$ from a DFA for L . Take a minimal DFA for L and for each state q , pick a string that reaches q from the start state. Let this set of strings be A . Do the same thing for a minimal DFA for L^R , the reversal of L , to get a set of strings B . This defines a set of contexts $A \times B^R$. Now pick one string from each element of the syntactic monoid of L to get a set of strings M . The Galois

connection between the set of contexts $A \times B^R$ and the set of strings M will be isomorphic to $\mathfrak{B}(L)$, from which it is straightforward to construct $\mathfrak{G}(L)$.

We will now give a complete example for a regular language. Consider $L = (ab)^*$. This has the 7 closed sets of strings: $\Sigma^*, \emptyset, L, \{\lambda\}, a(ba)^*, b(ab)^*, (ba)^*$. We therefore have 7 nonterminals in $\mathfrak{G}(L)$ that we denote $\top, \perp, S, \lambda, A, B, R$. The lexical rules in the grammar are: $\top \rightarrow a, \top \rightarrow b, \top \rightarrow \lambda, L \rightarrow \lambda, R \rightarrow \lambda, A \rightarrow a, B \rightarrow b$. The start symbol is S . There are 7^3 possible binary rules of which the following are in the grammar:

- Any production of the form $\top \rightarrow XY$
- Any production of the form $X \rightarrow \perp Y$ or $X \rightarrow Y \perp$.
- Any production of the form $X \rightarrow \lambda X$ or $X \rightarrow X \lambda$
- $L \rightarrow AB|LL$
- $A \rightarrow AR|LA$
- $B \rightarrow BL|RB$
- $R \rightarrow BA|RR$

$\mathfrak{G}(L)$ is a large but finite CFG. For any (binarised) grammar G that defines the same language, ϕ_L is an exact morphism from G to $\mathfrak{G}(L)$. That is to say this grammar contains a morphic image of every grammar for the language; this is analogous to the Universal Automaton, which we discuss now.

7.1 The Universal Automaton

We now consider the relation of the syntactic concept lattice to the universal automaton (UA) [Con71, LS⁺08]. We recall the definition of the UA using the terminology of [LS⁺08]; interestingly the Galois lattice structure of the UA does not form part of the standard presentations of the UA.

Definition 36 Given a language L , a factorisation of L is a maximal pair of sets of strings P, Q such that $PQ \subseteq L$.

We can of course consider this as a Galois connection between the prefixes and suffixes.

Definition 37 For a set of strings X , define $X^\blacktriangleleft = \{y | Xy \subseteq L\}$ and $X^\blacktriangleright = \{y | yX \subseteq L\}$.

These are the left and right residuals. A left factor is a set X (of prefixes) such that it is left-closed: $X = X^\blacktriangleleft$; conversely a right factor is one where $X = X^\blacktriangleright$. A factorisation (P, Q) is thus a pair where $P^\blacktriangleleft = Q$ and $Q^\blacktriangleright = P$; clearly P is left-closed and Q is right-closed.

Definition 38 For a language L , let $\text{UA}(L)$ be the set of all factorisations of L .

Definition 39 The Universal Automaton of a regular language L is defined to be a finite automaton where the states are factorisations, and where the set of initial states is $\{(P, Q) | \lambda \in P\}$, the set of final states is $\{(P, Q) | \lambda \in Q\}$, and the set of transitions is defined as $\{(P, Q) \xrightarrow{a} (P', Q') | PaQ' \subseteq L\}$.

We now consider a number of close relationships between the UA and the SCL for a given language.

Proposition 56 *Let (P, Q) be a factorisation of a language. Then both P and Q are closed sets of strings, in the lattice $\mathfrak{B}(L)$. i.e. $P = P^{\triangleright\triangleleft}$ and $Q = Q^{\triangleright\triangleleft}$.*

Proof If P is left-closed, then it is closed since $P = \{(\lambda, P^{\blacktriangleleft})\}^{\triangleleft}$. Similarly if Q is right closed, then $Q = \{(Q^{\blacktriangleright}, \lambda)\}^{\triangleleft}$.

However it is not necessarily the case that (P, λ) or (λ, Q) are closed sets of contexts.

Finally we note that the set of contexts of the empty string are equivalent to the factorisations.

Proposition 57

$$\{\lambda\}^{\triangleright} = \bigcup_{(P,Q) \in UA(L)} P \times Q$$

Proof Suppose (P, Q) is a factorisation; Therefore $P \times Q \odot \lambda = PQ \subseteq L$, so $P \times Q \subseteq \{\lambda\}^{\triangleright}$. Conversely if (l, r) in $\{\lambda\}^{\triangleright}$, then $lr \in L$ so (l, r) must be an element of at least one maximal factorisation.

8 Discussion

The SCL appears quite naturally under several equivalent definitions; as the Galois connection between strings and contexts; as the set of maximal decompositions of the language into strings and contexts, and as the unique minimal complete idempotent semiring that recognises a language. It can be seen as the lifting of the syntactic congruence to sets of strings rather than strings. It forms the basis for various algorithms for grammar induction. It also has deep relations with the syntactic congruence and the universal automaton. These considerations suggest that it is worthy of sustained analysis and study, as it potentially can give rise to new algorithms for various problems associated with context-free grammars.

Additionally realising that these structures are also residuated lattices, has as a consequence that they can be used to give a complete and sound semantics for the Lambek calculus [Cla12, Wur12]. This gives a new perspective on the slightly troubled relationship between Lambek grammars and context-free grammars; the SCL is a point at which these two formalisms seems to make contact in a way very similar to Lambek's original conception. Lambek justifies the rules of his calculus by appeal to the soundness of the deduction system with respect to residuated lattices and later says [Lam58]

We shall assign type n to all expressions which can occur in any context in which all proper names can occur.

If we write N for the set of all proper names, then the set of contexts in which all proper names can occur is N^\triangleright , and so this states, in our terminology, that the type n should be assigned to the set $N^{\triangleright\triangleleft}$. Thus the types of the Lambek calculus were, it appears, intended to be closed sets of strings.

The phrase “an algebraic theory of context-free languages” and indeed the now rather dated term “algebraic language” for context free language, has been used in a number of different contexts [CS63], as part of a program to base language theory not on phrase structure grammars or on machine models (finite state automata or push-down automata) but rather on some algebras (e.g. [Hot86]). The work we present here can be considered as part of the same research program; rather than dealing directly with the productions of a context-free grammar, we operate with an algebraic structure that interprets it; a complete idempotent semiring. Homomorphisms of this structure can then be used to model transformations, i.e. morphisms, of the original grammar. We then naturally arrive at a unique object for each grammar, the terminal element of a category, and thus a universal morphism for CFGs. This gives us a set of syntactic categories that can be used as nonterminals in a grammar; under the assumption that we want the grammar to be minimal in a fairly obvious sense, we can without loss of generality assume that the grammar uses only these nonterminals. Each production then has an easily interpretable form as an inequality (or equality) between these syntactic categories.

Though the computational problems of constructing the syntactic concept lattice from a context-free grammar is in general undecidable, it is straightforward to approximate it. By picking some finite set of contexts and finite set of substrings, and using a standard parser, we can efficiently compute a fragment of the lattice. This process relies only on information about the set of strings in the language and can thus be used as the basis for a variety of learning algorithms. Nonterminals in the grammar are defined which correspond to specific concepts in the lattice of the language being learned. The algorithm can either pick a finite set of strings S , which defines the concept $\mathcal{C}(S)$, or a finite set of contexts C , which defines the concept $\mathcal{C}(C^\triangleleft)$. Algorithms that use the former assumption are called *primal* and ones that use the latter are called *dual*. Given a suitable set of nonterminals defined primally, dually or using a combination of the two, the algorithm can then define a set of rules, which, given the objective nature of the nonterminals, is straightforward. Given nonterminals N, P, Q that correspond to concepts $\mathcal{N}, \mathcal{P}, \mathcal{Q}$ we want a production $N \rightarrow PQ$ if and only if $\mathcal{N} \geq \mathcal{P} \circ \mathcal{Q}$. Testing this inequality depends on how the nonterminals are defined, but in general the validity of the inequality depends only on the concepts used in the inequality. This means that we replace the problem of testing the global validity of an entire grammar with a test which is local and computationally tractable. This massively simplifies the process of constructing a grammar for a language, and allows for efficient learning algorithms under a number of different models [Cla10d, Yos11, Yos12].

The UA has been used as the basis for algorithms for minimising regular grammars; for finding the smallest nondeterministic automaton (NFA) that generates a given regular language. Though the problem of finding the min-

imal NFA that generates the same language as a given DFA is known to be PSPACE-complete [JR93], various algorithms have been proposed using the UA as a search space. Although the problem of finding the minimal CFG that generates the same language as a given CFG is clearly undecidable, we could use the SCL to find the smallest CFG that generates a given regular language, presented as an automaton, using an exhaustive search through the SCL.

This has a number of implications also for linguistics, where phrase structure grammars, of which context-free grammars are the simplest type, are widely used. Classically we identify the parse tree generated by the phrase structure grammar with the syntactic structure of the sentence, and the syntactic categories correspond therefore to the nonterminals of the grammar. From the perspective taken here, the minimality of grammars whose nonterminals are syntactic concepts is a very strong argument in favour of the use of syntactic concepts as syntactic categories in linguistic description. To rebut this argument would require an argument that the correct grammar is redundant in the sense that it contains mergeable nonterminals. In the absence of such an argument we can conclude, that at least in the domain of context-free grammars, that syntactic structure must supervene on distributional structure; which gives a strong in principle argument for the central importance of distributional analysis in linguistic theory.

Acknowledgments

I am very grateful to Ryo Yoshinaka and Hans Leiss for extensive technical discussions on the syntactic concept lattice, which have greatly improved, corrected and simplified the presentation; any remaining errors are of course my own. I am also especially grateful to Hans Leiss for pointing out the connection to compete idempotent semirings. I would also like to thank the anonymous reviewers for helpful suggestions.

References

- Ang82. D. Angluin. Inference of reversible languages. *Journal of the ACM*, 29(3):741–765, 1982.
- Ang87. D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987.
- BHKL09. B. Bollig, P. Habermehl, C. Kern, and M. Leucker. Angluin-style learning of NFA. In *Proceedings of IJCAI 21*, 2009.
- CE07. Alexander Clark and Rémi Eyraud. Polynomial identification in the limit of substitutable context-free languages. *Journal of Machine Learning Research*, 8:1725–1745, August 2007.
- Cla10a. Alexander Clark. Distributional learning of some context-free languages with a minimally adequate teacher. In José Sempere and Pedro Garcia, editors, *Grammatical Inference: Theoretical Results and Applications. Proceedings of the International Colloquium on Grammatical Inference*, pages 24–37. Springer-Verlag, September 2010.

- Cla10b. Alexander Clark. Efficient, correct, unsupervised learning of context-sensitive languages. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 28–37, Uppsala, Sweden, July 2010. Association for Computational Linguistics.
- Cla10c. Alexander Clark. Learning context free grammars with the syntactic concept lattice. In José Sempere and Pedro Garcia, editors, *Grammatical Inference: Theoretical Results and Applications. Proceedings of the International Colloquium on Grammatical Inference*, pages 38–51. Springer-Verlag, 2010.
- Cla10d. Alexander Clark. Three learnable models for the description of language. In Carlos Martín-Vide Adrian-Horia Dediu, Henning Fernau, editor, *Language and Automata Theory and Applications, Fourth International Conference, LATA 2010*, LNCS, pages 16–31. Springer-Verlag, 2010.
- Cla11. Alexander Clark. A learnable representation for syntax using residuated lattices. In Philippe Groote, Markus Egg, and Laura Kallmeyer, editors, *Formal Grammar*, volume 5591 of *Lecture Notes in Computer Science*, pages 183–198. Springer Berlin Heidelberg, 2011.
- Cla12. Alexander Clark. Logical grammars, logical theories. In Denis Bechet and Alexandre Dikovsky, editors, *Logical Aspects of Computational Linguistics*, pages 1–20. Springer, 2012.
- Con71. J.H. Conway. *Regular algebra and finite machines*. Chapman and Hall, London, 1971.
- CS63. N. Chomsky and M.P. Schützenberger. The algebraic theory of context-free languages. *Studies in Logic and the Foundations of Mathematics*, 35:118–161, 1963.
- DLT04. F. Denis, A. Lemay, and A. Terlutte. Learning regular languages using RFSAs. *Theoretical Computer Science*, 313(2):267–294, 2004.
- DP02. B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2002.
- Eil74. S. Eilenberg. *Automata, languages, and machines*. Academic press, 1974.
- ÉL05. Z. Ésik and H. Leiß. Algebraically complete semirings and Greibach normal form. *Annals of Pure and Applied Logic*, 133(1):173–203, 2005.
- GJKO07. N. Galatos, P. Jipsen, T. Kowalski, and H. Ono. *Residuated lattices: an algebraic glimpse at substructural logics*. Elsevier, 2007.
- Gol99. J.S. Golan. *Semirings and their Applications*. Springer, 1999.
- GR62. S. Ginsburg and H.G. Rice. Two families of languages related to ALGOL. *Journal of the ACM (JACM)*, 9(3):350–371, 1962.
- GW97. B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, 1997.
- Hot86. G. Hotz. A representation theorem of infinite dimensional algebras and applications to language theory. *Journal of Computer and System Sciences*, 33(3):423–455, 1986.
- Jip04. Peter Jipsen. From semirings to residuated Kleene lattices. *Studia Logica: An International Journal for Symbolic Logic*, 76(2):pp. 291–303, 2004.
- JR93. Tao Jiang and Bala Ravikumar. Minimal nfa problems are hard. *SIAM Journal on Computing*, 22(6):1117–1141, 1993.
- Lam58. J. Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65(3):154–170, 1958.
- Lei12. Hans Leiss. Learning CFGs with the finite context property: A note on A.Clark’s algorithm. Manuscript, 2012.
- Lom07. Sylvain Lombardy. On the size of the universal automaton of a regular language. In Wolfgang Thomas and Pascal Weil, editors, *STACS 2007*, volume 4393 of *Lecture Notes in Computer Science*, pages 85–96. Springer Berlin Heidelberg, 2007.
- LS⁺08. Sylvain Lombardy, Jacques Sakarovitch, et al. The universal automaton. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and automata: History and Perspectives*, volume 2, pages 457–504. Amsterdam University Press, Amsterdam, NL, 2008.
- Pol01. L. Polák. Syntactic semiring of a language. In *Mathematical Foundations of Computer Science 2001*, pages 611–620. Springer, 2001.

-
- Pol03. L. Polák. Syntactic semiring and universal automaton. In *Developments in language theory*, pages 162–162. Springer, 2003.
- PS84. F.C.N. Pereira and S.M. Shieber. The semantics of grammar formalisms seen as computer languages. In *Proceedings of the 10th International Conference on Computational Linguistics and 22nd annual meeting on Association for Computational Linguistics*, pages 123–129. Association for Computational Linguistics, 1984.
- RS59. M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM journal of research and development*, 3(2):114–125, 1959.
- Sak09. J. Sakarovitch. *Elements of automata theory*. Cambridge Univ Pr, 2009.
- Wur12. Christian Wurm. Completeness of full Lambek calculus for syntactic concept lattices. In *Proceedings of the 17th conference on Formal Grammar 2012 (FG)*, 2012.
- Yos11. Ryo Yoshinaka. Towards dual approaches for learning context-free grammars based on syntactic concept lattices. In Giancarlo Mauri and Alberto Leporati, editors, *Developments in Language Theory*, volume 6795 of *Lecture Notes in Computer Science*, pages 429–440. Springer Berlin Heidelberg, 2011.
- Yos12. Ryo Yoshinaka. Integration of the dual approaches in the distributional learning of context-free grammars. In *LATA*, pages 538–550, 2012.